

AD-A054 602

DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC F/G 12/1
AN INVESTIGATION OF THREE COMPUTER PROGRAMS FOR THE SOLUTION OF--ETC(U)
APR 78 D A GIGNAC
DTNSKDC-78/033

UNCLASSIFIED

NL

| OF |
AD
A054602

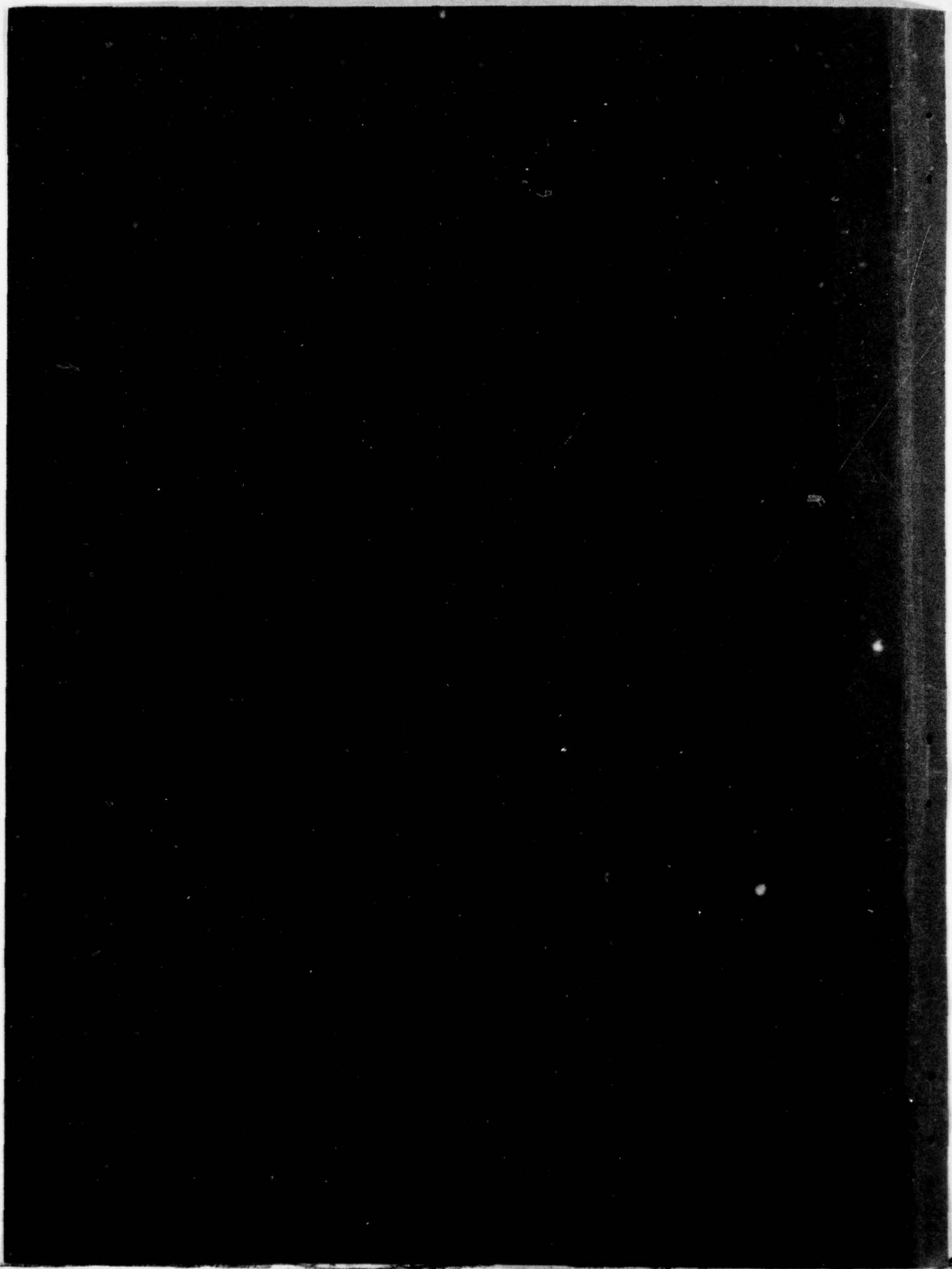
12
12
12



END
DATE
FILMED
6-78
DDC

AD No. _____
CDC FILE COPY

AD A 054602



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER DTNSRDC-78/833	2. GOVT ACCESSION NO. Programs	3. RECIPIENT'S CATALOG NUMBER 9
4. TITLE (and Subtitle) AN INVESTIGATION OF THREE COMPUTER PROGRAMS FOR THE SOLUTION OF $AX = B$ WHERE A IS SYMMETRIC AND SPARSE.	5. TYPE OF REPORT & PERIOD COVERED Interim Report. Jun 1977 - Feb 1978	
7. AUTHOR(s) Donald A. Gignac	8. CONTRACT OR GRANT NUMBER(s) SR01403, F53532	
9. PERFORMING ORGANIZATION NAME AND ADDRESS David W. Taylor Naval Ship Research and Development Center Bethesda, Maryland 20084	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS SR 0140301, ZF53532001 Work Units 1-1808-010, 1-1808-009	
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE Apr 1978	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 1239p.	13. NUMBER OF PAGES 41	
15. SECURITY CLASS. (of this report) UNCLASSIFIED		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Matrix Analysis Linear Equations Sparse Matrices Symmetric Matrices Triangular Decomposition		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report documents an investigation of three recent programs for the solution of $AX = B$ where A is symmetric and sparse: the Yale Sparse Matrix Package; the Munksgaard subroutines; and the Mesztenyi-Rheinboldt subroutines. The first two programs compute in-core solutions; the third uses random access to obtain its solution. The performance of these three programs is (Continued on reverse side)		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

387682

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

(Block 20 continued)

compared with that of a previously developed out-of-core equation solver, CSKYDG2. The two in-core equation solvers (especially the first) are faster than CSKYDG2; the third is slower. All three provide the same degree of accuracy as CSKYDG2, but they require large amounts of core storage. It would appear that the two in-core equation solvers are not suited for use on the CDC 6000 series of computers in their present form due to the limited amount of core storage available. Although the third equation solver does not require as much core storage, it does not perform as well as existing out-of-core equation solvers which use the same or lesser amounts of core storage.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

TABLE OF CONTENTS

	Page
ABSTRACT.....	1
INTRODUCTION.....	1
THE PROGRAMS.....	3
THE TEST EXAMPLES.....	4
TABLE NOTATION.....	5
OBSERVATIONS AND CONCLUSIONS.....	8
ACKNOWLEDGMENTS.....	9
PROGRAM LISTINGS.....	11
REFERENCES.....	35

LIST OF TABLES

- 1 - Execution Times for the First Set of Systems
 $A_N^1 X = B_N$ (CDC 6600 CPU Seconds)..... 6

- 2 - Execution Times for the Second Set of Systems
 $A_N^2 X = C_N$ (CDC 6600 CPU Seconds)..... 7

ACCESSION for		
RTIS	Write Section	<input checked="" type="checkbox"/>
DDO	Diff Section	<input type="checkbox"/>
UNANNOUNCED		<input type="checkbox"/>
JUSTIFICATION		
BY		
DISTRIBUTION/AVAILABILITY CODES		
FILE AVAIL. and/or SPECIAL		
A		

ABSTRACT

This report documents an investigation of three recent programs for the solution of $AX = B$ where A is symmetric and sparse: the Yale Sparse Matrix Package; the Munksgaard subroutines; and the Mesztenyi-Rheinboldt subroutines. The first two programs compute in-core solutions; the third uses random access to obtain its solution. The performance of these three programs is compared with that of a previously developed out-of-core equation solver, CSKYDG2. The two in-core equation solvers (especially the first) are faster than CSKYDG2; the third is slower. All three provide the same degree of accuracy as CSKYDG2, but they require large amounts of core storage. It would appear that the two in-core equation solvers are not suited for use on the CDC 6000 series of computers in their present form due to the limited amount of core storage available. Although the third equation solver does not require as much core storage, it does not perform as well as existing out-of-core equation solvers which use the same or lesser amounts of core storage.

INTRODUCTION

One of the long range projects of the Computation, Mathematics, and Logistics Department has been the development of mathematical subroutines suitable for use in the computer-aided structural analysis of ships. Many unrelated efforts in both government and industry have resulted in computer

programs that treat particular classes of structural problems. These programs often involve the solution of similar mathematical problems but, since the solutions are reached independently, the efficiency and accuracy of the various algorithms used may vary greatly. The need to coordinate these diverse efforts, to develop improved methods of more general applicability, and to produce more comprehensive programs for solving Navy structural problems became obvious. A project was therefore established to coordinate research efforts involving mathematical and computational methods in the area of structural mechanics and to integrate the work of mathematicians, computer specialists, and structural engineers in this field.

The present considerable interest in the finite element approach to structural analysis is evidenced by the widespread use of NASTRAN (Nasa STRuctural ANalysis program) and other such programs. According to the NASTRAN Theoretical Manual,^{1*} "From a theoretical viewpoint, the formulation of a static structural problem for solution by the displacement method is completely described by the matrix equation $KU=P$." Thus, there is a need for accurate efficient computer sub-routines capable of solving these large sparse positive definite systems of simultaneous linear equations. However, the order of K is often so large that, even when advantage is taken of K 's symmetry and banded structure, it is not feasible and sometimes not even possible, to store K in the core memory of a computer.

This report compares three recently developed programs for solving $KU=P$, where K is a matrix of very large order, with one of the author's previously developed out-of-core equation solvers.

* A complete listing of references is given on page 35.

THE PROGRAMS

The following programs are considered in this report:

- The Yale Sparse Matrix Package^{2,3} is a collection of subroutines which can be used to solve both symmetric and non-symmetric systems. It uses algorithms developed by Eisenstat, et al.^{2,3} This package is to some extent proprietary since limitations are placed on its use and distribution.
- The Munksgaard collection of subroutines⁴ implements Duff's algorithm for solving sparse symmetric systems. It can obtain a stable decomposition of K in both the definite and indefinite cases.⁵ This program was developed at the Technical University of Denmark and the Atomic Energy Research Establishment of the United Kingdom.
- The Mesztenyi-Rheinboldt package of subroutines^{6,7} can be used to solve both symmetric and non-symmetric systems by means of triangular decomposition. These subroutines are intended for use with systems whose coefficient matrix K will fit into core but may not do so after decomposition. Some of these subroutines were modified by the present author to more efficiently utilize the random access storage capabilities of the CDC 6000 series of computers.
- CSKYDG2 is an out-of-core Cholesky algorithm equation solver developed by the present author.⁸ It makes use of auxiliary storage via the random access capabilities of the CDC 6000 series of computers.

THE TEST EXAMPLES

EXAMPLE 1

The matrix family of Table 1, $A_{N^2}^1$, is generated as follows: Let N be an integer ≥ 3 . Let C_N be the tridiagonal of order N with 4's on the diagonal and a line of -1's above and below the diagonal. Let I_N be the identity matrix of order N . An $(N+1)$ -banded matrix of order N^2 , A_N^1 is constructed by

1. stringing N C_N submatrices along the diagonal,
2. inserting lines of $N-1$ $-I_N$ submatrices above and below the diagonal, and
3. setting the remaining elements of $A_{N^2}^1$ equal to 0.

Application of Gerschgorin's theorem shows $A_{N^2}^1$ to be positive definite. The right-hand side of the system $A_{N^2}^1 X = B_{N^2}$ is chosen such that all components of the exact solution vector except the first, which is 1, have the value 0.

EXAMPLE 2

The matrix family of Table 2, $A_{N^2}^2$, is generated as follows: Let N be an integer ≥ 3 . Assume the real symmetric matrix of order N^2 and bandwidth N with N^2 on the diagonal and -1 elements filling out the rest of the band. Then change the value of each zero element in the last N rows and columns to a -1. As before, Gerschgorin's theorem shows $A_{N^2}^2$ to be positive definite. Note that from the viewpoint of bandwidth, $A_{N^2}^2$ is a full matrix. The right-hand side of the system $A_{N^2}^2 X = D_{N^2}$ is chosen such that all the components of the exact solution vector, except the first, which is 1, are zero.

TABLE NOTATION

The solution times for $A_N^1 X = B_N$ and $A_N^2 X = D_N$ are tabulated in Tables 1 and 2, respectively. The CSKYDG2 times were originally run on the CDC 6600 at DTNSRDC.⁸ The other programs were run on the CDC 6400 at DTNSRDC and the resulting times were divided by 3 to give the equivalent CDC 6600 times.

The column headings are defined as follows:

- N - the order of the system
- M - system bandwidth
- T_{ORDER} - the time required for the ORDV subroutine from the Yale package to reorder the system
- T_{SOLVE} - the time required for either the SDRV subroutine from the Yale package or CSKYDG2 to factor and solve the system
- T_{TOTAL} - the total time required by a program to solve a system
- T_{DECOMP} - the time required for either the INDANL subroutine of the Munksgaard program or the SDEC01 subroutine of the Mesztenyi-Rheinboldt program to factor the coefficient matrix
- $T_{BACKSUP}$ - the time required for either the INDOPR subroutine of the Munksgaard program or the SSLV subroutine of the Mesztenyi-Rheinboldt program to solve the factored system
- T_{SETUP} - the time required by the SETUP preprocessor subroutine for CSKYDG2

(The above times are given in terms of CDC 6600 CPU seconds.)

TABLE 1 - EXECUTION TIMES FOR THE FIRST SET OF SYSTEMS $A_N^1 X = B_N$
(CDC 6600 CPU SECONDS)

N	M	YALE Execution Times			MUNKSGAARD Execution Times			CSKYDG2 Execution Times			M-R Execution Times		
		T _{ORDER}	T _{SOLVE}	T _{TOTAL}	T _{DECOMP}	T _{BACKSUB}	T _{TOTAL}	T _{SETUP}	T _{SOLVE}	T _{TOTAL}	T _{DECOMP}	T _{BACKSUB}	T _{TOTAL}
25	6	0.029	0.015	0.044	0.028	0.002	0.03	0.04	0.05	0.09	0.031	0.011	0.042
100	11	0.156	0.093	0.249	0.23	0.005	0.24	0.45	0.22	0.67	0.449	0.058	0.507
225	16	0.044	0.306	0.350	0.853	0.020	0.873	2.20	1.17	3.37	2.544	0.141	2.685
400	21	0.933	0.675	1.018	2.323	0.041	2.364	6.77	2.12	8.89	9.357	0.275	9.632
625	26	1.69	1.45	3.14	5.255	0.068	5.323	16.39	5.64	22.03	33.603	0.373	33.976
900	31	2.81	2.64	5.45	9.717	0.105	9.822	33.71	8.29	42.00			
1225	36	4.00	0.555	4.555	15.194	0.133	15.327	62.86	17.11	79.97			
1600	41				30.158	0.194	30.352	106.49	21.96	128.45			
2025	46							107.75	39.88	210.63			
2500	51							259.74	49.20	308.94			

TABLE 2 - EXECUTION TIMES FOR THE SECOND SET OF SYSTEMS $A_N^2 X = C_N$
(CDC 6600 CPU SECONDS)

N	YALE Execution Times			MINKSGAARD Execution Times			CSKYDG2 Execution Times			M-R Execution Times		
	T _{ORDER}	T _{SOLVE}	T _{TOTAL}	T _{DECOMP}	T _{BACKSUB}	T _{TOTAL}	T _{SETUP}	T _{SOLVE}	T _{TOTAL}	T _{DECOMP}	T _{BACKSUB}	T _{TOTAL}
25	0.08	0.022	0.102	0.046	0.003	0.049	0.04	0.06	0.10	0.048	0.012	0.060
100	0.517	0.189	0.706	0.635	0.011	0.646	0.47	0.45	0.92	1.058	0.068	1.126
225	2.292	0.745	3.037	3.183	0.037	3.220	2.22	2.38	4.60	8.027	0.192	8.219
400	2.04	2.04	4.08	10.278	0.083	10.361	6.89	5.22	12.11	33.589	0.416	34.005
625	2.264	1.948	4.212				16.68	13.99	30.67			
900	2.402	2.976	5.378				34.28	23.10	57.38			
1225							63.56	47.38	110.94			
1600							108.83	67.55	176.38			
2025							173.70	118.55	292.25			
2500							264.18	157.16	421.34			

OBSERVATIONS AND CONCLUSIONS

An examination of the times in Tables 1 and 2 indicates that the Yale program is the fastest by far of all four programs, ranging from several times up to 15 times faster than CSKYDG2 (total time). The Munksgaard program is at best some 4 to 5 times faster than CSKYDG2 (total time) for example 1 but just noticeably faster than CSKYDG2 (total time) for example 2. Why the Yale program is so much faster than the Munksgaard program is not immediately clear.

The Mesztenyi-Rheinboldt program appears to be somewhat slower than CSKYDG2 because CSKYDG2 moves several rows at one time in out of auxiliary core storage whereas the present version of the Mesztenyi-Rheinboldt program moves only one row at a time. (As a matter of fact the UNIVAC version of the Mesztenyi-Rheinboldt program moves individual elements. This was changed to row movement by the present author when converting the program to the CDC 6000 series of computers.)

The Yale and Munksgaard programs have a tremendous time advantage over the Mesztenyi-Rheinboldt program and CSKYDG2 because the first two programs do everything in core while the latter two programs use random access to move rows in and out of auxiliary storage. However, a field length of 300000 CM was required to obtain the times given for the Yale and Munksgaard programs in Tables 1 and 2 even though the simplest of driving programs was used. (The Mesztenyi-Rheinboldt times required a field length of 225000 CM.) Clearly the use of some device such as "overlay" would be required to make use of either the Yale or the Munksgaard program in some applications. The Yale and Munksgaard programs are really suited for computers such as the Texas Instruments' Advanced Scientific Computer which have abundant core storage, although the programs would have to be rewritten for the most part to obtain the full benefit of the

computer's optimizing capability. For further testing on the CDC 6000 series these programs should definitely be modified by the introduction of integer packing and unpacking sub-routines to save core storage by cutting down the size of the integer arrays.

ACKNOWLEDGMENTS

The author wishes to thank the following individuals for their interest and assistance: Dr. Elizabeth H. Cuthill (DTNSRDC Code 1805); Mrs. Sharon Good (DTNSRDC Code 1892); Dr. Gordon Everstine and Mr. Michael Golden (DTNSRDC Code 1844); Mr. Richard Van Eseltine and Mr. Paul Morawski (DTNSRDC Code 1843); and Mrs. Barbara Brooks (NRL Code 422.23).

PROGRAM LISTINGS

Listings are provided for the Munksgaard and Mesztenyi-Rheinboldt programs since it was necessary to obtain the times in Tables 1 and 2. Listings for CSKYDG2 and the Yale program will be found in Gignac⁸ and Eisenstat et al.,² respectively. As noted previously, the Yale Sparse Matrix Package is considered proprietary in some sense.

THE MESZTENYI-RHEINBOLDT PROGRAM

```

SUBROUTINE SINT01
*   COMMON MD(8),FD(7),RY(10000),CY(10000),A(10000),AN(1600),ND(1600),
COMMON MD(8),FD(7),RY(23000),CY(23000),A(23000),AN(1600),ND(1600),
1 IE(1600),IH(1600),IP(1600),INDEX(1601),B(1600),X(1600)
INTEGER RY,CY
C *****
C * INITIALIZE SYMMETRIC STRUCTURE AND COEFFICIENT ARRAYS *
C *****
C
N=MD(1)
MD(5)=N
FD(3)=0.
DO 10 I=1,N
AN(I)=0.
RY(I)=I
CY(I)=I
10 ND(I)=1
RETURN
END

SUBROUTINE SBLD01(I,J,V)
*   COMMON MD(8),FD(7),RY(10000),CY(10000),A(10000),AN(1600),ND(1600),
COMMON MD(8),FD(7),RY(23000),CY(23000),A(23000),AN(1600),ND(1600),
1 IE(1600),IH(1600),IP(1600),INDEX(1601),B(1600),X(1600)
INTEGER RY,CY
C *****
C * BUILD SYMMETRIC STRUCTURE AND COEFFICIENT ARRAYS *
C *****
C
S=V*V
FD(3)=AMAX1(FD(3),ABS(V))
AN(I)=AN(I)+S
IF (I.EQ.J) GO TO 20
MD(5)=MD(5)+1
M0=MD(5)
A(M0)=V
AN(J)=AN(J)+S
ND(I)=ND(I)+1
ND(J)=ND(J)+1
IF (I.GT.J) GO TO 10
RY(M0)=RY(I)
CY(M0)=CY(J)
RY(I)=M0
CY(J)=M0
RETURN
10 RY(M0)=RY(J)
CY(M0)=CY(I)
RY(J)=M0
CY(I)=M0
RETURN
20 A(I)=V
RETURN
END

```

BEST AVAILABLE COPY

```

SUBROUTINE SDEC01
COMMON MD(8),FD(7),RY(23000),CY(23000),A(23000),AN(1600),ND(1600),
1 IE(1600),IH(1600),IP(1600),INDEX(1601),B(1600),X(1600)
INTEGER RY,CY
DIMENSION BB(1600)
COMMON /ONE/ LIM1(1600)
C *****
C * DECOMPOSE SYMMETRIC MATRIX *
C *****
C
      N=MD(1)
      MD(4)=0
      FD(5)=0.
      FD(6)=1.
      FD(7)=0.
      FD(4)=0.
      NM=0
      MD(6)=MD(5)
      MD(7)=MD(5)
      MD(8)=0
      DO 10 I=1,N
      FD(7)=FD(7)+ALOG(AN(I))
10 IP(I)=I
      FD(7)=0.5*FD(7)
      CALL SVMI
*
C
C LOOP ON PIVOTING
C
      DO 250 I=1,N
      K=0
      IF (I.EQ.N) GO TO 30
C
C SELECT PIVOT BY MINIMAL DEGREE
C
      NOX=N+1
      AX=0.
      DO 15 J=I,N
      IX=IP(J)
15 AX=AMAX1(AX,ABS(A(IX)))
      AX=AX*FD(2)
      DO 20 J=I,N
      IX=IP(J)
      IF (ABS(A(IX)).LT.AX) GO TO 20
      IF (ND(IX).GE.NOX) GO TO 20
      NOX=ND(IX)
      IY=J
20 CONTINUE
      IF (I.EQ.IY) GO TO 30
      J=IP(IY)
      IP(IY)=IP(I)
      IP(I)=J
C
C COLLECT THE ROW AND COLUMN OF THE PIVOT
C ALSO DELETE THEM FROM THE STORAGE
C
30 IX=IP(I)
      S=A(IX)

```

BEST AVAILABLE COPY

```

      IF (ABS(S).LT.FD(1)) GO TO 300
      IF (I.EQ.N) GO TO 110
      IV1=0
      IY=IX
40    IY=RY(IY)
      IF (IY.EQ.IX) GO TO 70
      IZ=IY
50    IZ=CY(IZ)
      IF (IZ.GT.N) GO TO 60
      K=K+1
      IE(K)=IY
      IM(K)=IZ
      AN(K)=A(IY)
      A(IY)=0.
      ND(IZ)=ND(IZ)-1
60    IF (CY(IZ).NE.IY) GO TO 50
      CY(IZ)=CY(IY)
      CY(IY)=MM
      MM=IY
      GO TO 40
70    IY=CY(IY)
      IF (IY.EQ.IX) GO TO 100
      IZ=IY
      IY1=IY
80    IZ=RY(IZ)
      IF (IZ.GT.N) GO TO 90
      K=K+1
      IE(K)=IY
      IM(K)=IZ
      AN(K)=A(IY)
      A(IY)=0.
      ND(IZ)=ND(IZ)-1
90    IF (RY(IZ).NE.IY) GO TO 80
      RY(IZ)=RY(IY)
      GO TO 70
100   IF (IY1.EQ.0) GO TO 110
      CY(IY1)=MM
      MM=CY(IX)

```

C
C MODIFICATION OF THE ROW ELEMENTS
C

```

110  FD(4)=AMAX1(FD(4),ABS(S))
      FD(5)=FD(5)+ALOG(ABS(S))
      IF (S.LT.0.) FD(6)=-FD(6)
*    CALL SVM(-IX,S)
      BB(1)=-IX
      BB(2)=S
      LIN=4
      MD(8)=MD(8)+1+K*K
*    IF (K.EQ.0) GO TO 250
      IF (K.EQ.0) GO TO 249
      DO 115 J=1,K
      AN(J)=AN(J)/S
*    CALL SVM(IM(J),AN(J))
      BB(LIN-1)=IM(J)
      BB(LIN)=AN(J)
      FD(4)=AMAX1(FD(4),ABS(AN(J)))

```

BEST AVAILABLE COPY

```

115 LIM=LIM+2
    K1=K-1
C
C LOOP FOR THE CROSS-POINT ELEMENTS
C
    DO 240 J=1,K
        J1=J+1
        IZ=IH(J)
        Z=AN(J)
        A(IZ)=A(IZ)-S*Z*Z
        FD(4)=AMAX1(FD(4),ABS(A(IZ)))
        IF (J.EQ.K) GO TO 240
        DO 230 JJ=J1,K
            JZ=IH(JJ)
            I1=MIN0(IZ,JZ)
            I2=MAX0(IZ,JZ)
            L1=RY(I1)
            L2=CY(I2)
120 IF ((L1.EQ.I1).OR.(L2.EQ.I2)) GO TO 140
            IF (L1.EQ.L2) GO TO 220
            IF (L1.GT.L2) GO TO 130
            L2=CY(L2)
            GO TO 120
130 L1=RY(L1)
            GO TO 120
C INSERTION OF A NON-ZERO ELEMENT
140 ND(I1)=ND(I1)+1
        ND(I2)=ND(I2)+1
        MD(7)=MD(7)+1
        IF (MM.NE.0) GO TO 170
C USE NEW STORAGE
        MD(6)=MD(6)+1
        IF (MD(6).GT.MD(2)) GO TO 310
160 L1=MD(6)
        A(L1)=0
        RY(L1)=RY(I1)
        CY(L1)=CY(I2)
        RY(I1)=L1
        CY(I2)=L1
        GO TO 220
C USE AVAILABLE STORAGE
170 L1=MM
        MM=CY(MM)
        L3=I1
        L2=I2
180 IF (RY(L3).LT.L1) GO TO 190
        L3=RY(L3)
        GO TO 180
190 RY(L1)=RY(L3)
        RY(L3)=L1
200 IF (CY(L2).LT.L1) GO TO 210
        L2=CY(L2)
        GO TO 200
210 CY(L1)=CY(L2)
        CY(L2)=L1
C WRITE OUT CROSS-POINT ELEMENT
220 A(L1)=A(L1)-AN(J)*AN(JJ)*S

```

BEST AVAILABLE COPY

```

230 FD(4)=AMAX1(FD(4),ABS(A(L1)))
240 CONTINUE
C END OF MODIFICATION LOOP
* 250 CALL SVN(-IX,S)
249 BB(LIN-1)=-IX
    BB(LIN)=S
    LINI(I)=LIN
250 CALL WRITMS(7,BB,LIN,I)
C
C END OF PIVOTING LOOP
C
*    CALL SVME
    RETURN
C
C SINGULAR MATRIX
C
300 MD(4)=1
    RETURN
310 MD(4)=3
    RETURN
C
END

```

BEST AVAILABLE COPY

```

SUBROUTINE SSLV(MD,X,Y)
  DIMENSION MD(1),X(1),Y(1)
  DIMENSION BB(1600)
  COMMON /ONE/ LIM(1600)
C * *****
C * BACKSUBSTITUTION FOR SYMMETRIC DECOMPOSED MATRIX *
C * *****
C
  N=MD(1)
  DO 10 I=1,N
    10 Y(I)=X(I)
  *   CALL SVRI
    I=0
C FORWARD SUBSTITUTION
  * 20 CALL SVRF(L2,Z)
    20 I=I+1
    CALL READMS(7,BB,LIM(I),I)
  *   L2=-L2
    L2=-BB(1)
    S=Y(L2)
    LIM=4
  * 30 CALL SVRF(L2,Z)
    30 L2=BB(LIM-1)
    Z=BB(LIM)
    IF (L2.LT.0) GO TO 40
    Y(L2)=Y(L2)-Z*S
    LIM=LIM+2
    GO TO 30
  40 IF (I.LT.N) GO TO 20
C BACKWARD BACKSUBSTITUTION
  * 50 CALL SVRB(L2,Z)
    *   J=-L2
    50 CALL READMS(7,BB,LIM(I),I)
    LIM=LIM(I)
    J=-BB(LIM-1)
    Z=BB(LIM)
    Y(J)=Y(J)/Z
    I=I-1
  * 60 CALL SVRB(L2,Z)
    60 LIM=LIM-2
    L2=BB(LIM-1)
    Z=BB(LIM)
    IF (L2.LT.0) GO TO 70
    Y(J)=Y(J)-Z*Y(L2)
    GO TO 60
  70 IF (I.GT.0) GO TO 50
  RETURN
C
  END

```

BEST AVAILABLE COPY

THE MUNKSGAARD PROGRAM

```

SUBROUTINE INDANL(A,IND1,IND2,NZ1,IA1,IA2,N,IW,IP,D,G,U,W)
* IMPLICIT REAL*8 (A-H,S-Z),INTEGER (I-R)
* IMPLICIT REAL (A-H,S-Z),INTEGER (I-R)
* DOUBLE PRECISION A(IA2),D(N),W(N)
* DIMENSION A(IA2),D(N),W(N)
* INTEGER IP(N,2),KP(2),KL(2),JP(2),FIL
* INTEGER*2 IW(N,7),IND1(IA1),IND2(IA2)
* INTEGER IW(N,7),IND1(IA1),IND2(IA2)
* LOGICAL FIRST,STAB1,STAB2
COMMON/INDEFI/FIL,NUEL,LROW,LCOL,NCP,NUCL
C IP(I,1),IP(I,2) POINT TO THE START OF ROW/COLUMN I.
C IW(I,1),IW(I,2) HOLD THE NUMBER OF NONZEROS IN ROW/COLUMN I OF THE
C LOWER TRIANGULAR PART OF A.
C DURING THE MAIN BODY OF THIS SUBROUTINE THE VECTORS IW(*,3),IW(*,4)
C IW(*,5) ARE USED TO HOLD DOUBLY LINKED LISTS OF ROWS THAT HAVE
C NOT BEEN PIVOTAL AND HAVE EQUAL NUMBER OF NONZEROS.
C IW(I,3) HOLD FIRST ROW/COLUMN TO HAVE I NONZEROS OR ZERO IF THERE
C ARE NONE.
C IW(I,4) HOLD ROW/COLUMN NUMBER OF ROW/COLUMN PRIOR TO ROW I IN ITS
C LIST OR ZERO IF NONE.
C IW(I,5) HOLD ROW/COLUMN NUMBER OF ROW/COLUMN AFTER ROW I IN ITS LIST
C OR ZERO IF NONE.
C IW(*,6) AND IW(*,7) ARE USED TO UNPAC THE PIVOT ROW(S) INVOLVED IN
C AN ACTUAL ROW OPERATION.
C DURING THE MAIN BODY OF THE SUBROUTINE IND1 AND IND2 KEEP A COLUMN
C FILE AND A ROW FILE CONTAINING RESPECTIVELY THE ROW NUMBERS OF
C THE NON-ZEROS OF EACH COLUMN AND THE COLUMN NUMBERS OF THE NON-
C ZEROS OF EACH ROW. THE NON-ZEROS OF A FOLLOWS THE ORDERING OF THE
C ROW FILE IND2. THE IP ARRAYS POINTS TO THE START POSITION IN IND2
C (AND A) AND IND1 OF EACH ROW AND COLUMN.
C
DO 5 I=1,N
DO 4 J=6,7
4 IW(I,J)=-1
D(I)=0.
W(I)=0.
DO 5 J=1,5
5 IW(I,J)=0
G=0.
NZ=NZ1
NUEL=NZ
L=1
C
C COUNT NUMBER OF ELEMENTS
DO 20 IDUMMY=1,NZ
IF (L.GT.NUEL) GOTO 25
DO 10 K=L,NUEL
I=IND1(K)
J=IND2(K)
IF (I.LT.1 .OR. I.GT.N) GOTO 610
IF (J.LT.1 .OR. J.GT.N) GOTO 610
* GI=DABS(A(K))
* GI=ABS(A(K))
G=DMAX1(GI,G)
G=AMAX1(GI,G)
IF (I.EQ.J) GOTO 15
IF (W(I).LT.GI)W(I)=GI

```

BEST AVAILABLE COPY

```

      IF (W(J).LT.GI)W(J)=GI
      IW(I,1)=IW(I,1)+1
10    IW(J,2)=IW(J,2)+1
      GOTO 25
C
C CHECK FOR DOUBLE ENTRIES ON THE DIAGONAL AND REMOVE DIAGONAL FORM
C THE FILE.
15    IR=I
      J=I
      IF (D(I).NE..0) GOTO 650
      D(I)=A(K)
      L=K
      A(L)=A(NUEL)
      IND1(L)=IND1(NUEL)
      IND2(L)=IND2(NUEL)
      IND1(NUEL)=0
      IND2(NUEL)=0
20    NUEL=NUEL-1
C
C MCP IS THE NUMBER OF COMPRESSES PERMITTED BEFORE A MORNING RESULTS.
C NCP IS THE NUMBER OF COMPRESSES.
25    NCP=0
      MCP=MAX0(N/10,20)
      NZ=NUEL
      LCOL=NUEL
      NUCL=NUEL
      LROW=NUEL
C
C CHECK FOR NULL ROW AND INITIALIZE IP(I,1) AND IP(I,2) TO POINT JUST
C BEYOND WHERE THE LAST COMPONENT OF ROW/COLUMN I OF A WILL BE STORED.
      KI=1
      KJ=1
      DO 30 I=1,N
      KI=KI+IW(I,1)
      KJ=KJ+IW(I,2)
      IP(I,1)=KI
      IP(I,2)=KJ
      IF (IW(I,1)+IW(I,2).LE.0 .AND. D(I).EQ.0.) GOTO 630
30    CONTINUE
C
C REORDER BY ROWS USING IN-PLACE SORT ALGORITHM.
      DO 50 I=1,NZ
      IR1=IND1(I)
C IF IR1 IS NEGATIVE THE ELEMENT IS IN PLACE ALREADY.
      IF (IR1.LT.0) GOTO 50
      IC1=IND2(I)
      A1=A(I)
      K1=IP(IR1,1)-1
      DO 45 IDUMMY=1,NZ
      IF (I.EQ.K1) GOTO 46
      IR2=IND1(K1)
      IC2=IND2(K1)
      A2=A(K1)
      A(K1)=A1
      IND1(K1)=-IR1
      IND2(K1)=IC1
      IP(IR1,1)=K1

```

BEST AVAILABLE COPY

```

      IR1=IR2
      IC1=IC2
      A1=A2
45  K1=IP(IR1,1)-1
46  IF (IDUMMY.EQ.1) GOTO 49
      A(I)=A1
      IND1(I)=-IR1
      IND2(I)=IC1
49  IP(IR1,1)=I
50  CONTINUE
C
C CHECK FOR DOUBBEL ENTRIES WHILE USING THE CONSTRUCTED ROW FILE TO SET
C UP THE COLUMN FILE.
      KLL=NZ
      DO 60 I=1,N
      IR=N+1-I
      KPP=IP(IR,1)
      IF (KPP.GT.KLL) GOTO 60
      DO 55 K=KPP,KLL
      J=IND2(K)
      IF (IW(J,4).EQ.IR) GOTO 650
      IW(J,4)=IR
      KR=IP(J,2)-1
      IP(J,2)=KR
55  IND1(KR)=IR
60  KLL=KPP-1
C
C SET UP LINKED LISTS OF ROWS/COLUMNS WITH EQUAL NUMBER OF NON-ZEROS.
      DO 62 I=1,N
      NZI=IW(I,1)+IW(I,2)+1
      IN= IW(NZI,3)
      IW(NZI,3)=I
      IW(I,5)=IN
      IW(I,4)=0
62  IF (IN.NE.0) IW(IN,4)=I
C
C START THE ELIMINATION LOOP
C
      PP=0
      GG=G
      DO 500 IIP=1,N
      IF(PP.NE.2) GOTO 70
      PP=0
      GOTO 500
70  MY1=(N-1)**2
      MY2=(2*N-4)*(N-2)
      MYH=MY2
      FIRST=.TRUE.
      STAB2=.FALSE.
C
C SEARCH ROWS WITH RJP1 NONZEROS.
      DO 145 RJP1=1,N
C
C MY1 AND MY2 ARE THE LEAST COSTS SO FAR FOR STABLE 1X1 AND 2X2 PIVOTS.
C MCH IS THE THE LEAST OBTAINABLE COST.
      MCH=MIN0((RJP1-1)**2, (2*RJP1-4)**2/2)
      IF (MYH.GT.MCH) GOTO 72

```

BEST AVAILABLE COPY

```

        IF (MY1.GT.MY2.OR.FIRST) GOTO 71
        GOTO 210
71  IF (.NOT. STAB2) GOTO 72
        GOTO 220
72  JJP=IW(RJP1,3)
C
C  LOOP ON ROWS OF RJP1 NONZEROS.
        DO 140 IDUMMY =1,N
        IF(JJP.LE.0) GOTO 145
        II1=0
        MYB=(2*N-4)**2/2
        IF (.NOT. FIRST) GOTO 75
*   AU=DABS(D(JJP))/U
        AU= ABS(D(JJP))/U
        STAB1=(W(JJP).LE.AU)
        IF (STAB1) GOTO 105
73  P2=0
C
C  COMPRESS ROWFILE IF THERE IS NOT ROOM FOR NEW ELEMENTS.
        IF (NUEL+RJP1.LT.IA2) GOTO 75
        IF (LKOW+RJP1.GT.IA2.OR.MCP.GE.MCP) GOTO 670
        CALL COMPRE(A,IND2,IA2,N,IW,IP,.TRUE.)
C
C  PERFORM THE 1X1 STABILITY TEST OF ROW/COLUMN JJP.
75  DO 86 L=1,2
        KPP=IP(JJP,L)
        KLL=KPP+IW(JJP,L)-1
        IF (KPP.GT.KLL) GOTO 86
        DO 85 K=KPP,KLL
        IF (L.EQ.2) GOTO 77
        PP=IND2(K)
        GOTO 78
77  PP=IND1(K)
78  RJP2=IW(PP,1)+IW(PP,2)+1
C
C  BUILD UP THE FULL ROW JJP AT THE END OF THE ROW FILE. OFF DIAGONAL
C  ELEMENTS WHICH HAS BEEN INVESTEGATED AS 2X2 PIVOT CANDIDATES ARE
C  SET TO THE NEGATION OF THEIR COLUMN NUMBER.
        IND2(IA2-II1)=-PP
        IF (RJP2.LT.RJP1) GOTO 85
        IF (RJP2.GT.RJP1) GOTO 83
        JLK=IW(JJP,4)
        DO 82 KDUHMY =1,N
        IF (JLK.EQ.0) GOTO 83
        IF (JLK.EQ.PP) GOTO 85
82  JLK=IW(JLK,4)
83  IND2(IA2-II1)=PP
        MCN=(RJP1+RJP2-4)**2/2
        IF (MCN.GE.MY2.OR.MCN.GE.MYB) GOTO 85
        MYB=MCN
        P2=PP
        KKP=K
85  II1=II1+1
86  CONTINUE
        MCN=MY2
        DO 104 JDUMMY=1,N
        IF (P2.LE.0) GOTO 110

```

BEST AVAILABLE COPY

```

C
C CHECK THE STABILITY OF THE 2X2 PIVOT DETERMINED BY ROW JJP AND P2.
  JP(1)=JJP
  JP(2)=P2
  II2=IA2
  AA1=D(JP(1))
  AA2=D(JP(2))
  IF (IND2(KKP).EQ.P2) GOTO 88
  KPP=IP(P2,1)
  KPL=KPP+IW(P2,1)-1
  DO 87 KKP=KPP,KPL
  IF (IND2(KKP).EQ.JP(1)) GOTO 88
87 CONTINUE
88 AA12=A(KKP)
  DETP=AA1*AA2-AA12**2
  IF (DETP.NE.0.) GOTO 91
  APU=1.0070
  GOTO 92
* 91 APU=DMAX1(DABS(AA1),DABS(AA2))
  91 APU=AMAX1(ABS(AA1),ABS(AA2))
* APU=(APU+DABS(AA12))*U/DABS(DETP)
  APU=(APU+ABS(AA12))*U/ABS(DETP)
C
C DETERMINE THE NUMERICAL MAXIMAL ELEMENT OF ROWS JP(1),AND JP(2)
C WHICH IS NOT CONTAINED IN THE BLOCK DIAGONAL.
* GI=DMAX1(W(JP(1)),W(JP(2)))
  GI=AMAX1(W(JP(1)),W(JP(2)))
  IF (APU*GI.GT.1) GOTO 92
  JP1=JJP
  JP2=P2
  KP12=KKP
  A11=AA1/DETP
  A22=AA2/DETP
  A12=AA12/DETP
  MY2=MY8
  STAB2=.TRUE.
  GOTO 110
92 PP2=0
  KB=0
  JPL=JJP
  DO 98 L=1,2
  KPP=IP(JPL,L)
  KLL=KPP+IW(JPL,L)-1
  IF (KPP.GT.KLL) GOTO 98
  DO 97 K=KPP,KLL
  PP=IND2(II2)
  IF (PP.EQ.JP(2)) GOTO 95
  IF (PP.LE.0) GOTO 96
  MC=(RJP1+IW(PP,1)+IW(PP,2)-3)**2/2
  IF (MC.GE.MCN) GOTO 96
  MCN=MC
  KB=K
  PP2=PP
  GOTO 96
95 IND2(II2)=-P2
96 II2=II2-1
97 CONTINUE

```

BEST AVAILABLE COPY

```

98  CONTINUE
    MYB=MCN
    MCN=MY2
    P2=PP2
104  KKP=KB
C
C IF THE FIRST STABLE 1X1 PIVOT IS FOUND STORE RELEVANT POINTERS.
105  MY1=(RJP1-1)**2
    JPP=JJP
    FIRST=.FALSE.
C
C DETERMINE WETHER THE SPARSITY CONDITION IS SATISFIED OR NOT.
110  MYM=MIND(MY1,MY2)
    IF (MYM.GT.MCN) GOTO 140
    IF (MY1.GT.MY2.OR.FIRST) GOTO 120
    GOTO 210
120  IF (.NOT. STAB2) GOTO 140
    GOTO 220
140  JJP=IW(JJP,5)
145  CONTINUE
C
C PIVOT IS FOUND
C
C ROW JP1 IS USED AS 1X1 PIVOT.
210  JP1=JPP
    PP=1
    GOTO 230
C
C ROWS JP1 AND JP2 ARE USED AS 2X2 PIVOT.
220  PP=2
    JP(2)=JP2
230  JP(1)=JP1
C
C REMOVE ROWS/COLUMNS INVOLVED IN ELIMINATION FROM ORDERING VECTORS.
DO 266 L1=1,PP
    JPL=JP(L1)
DO 250 L=1,2
    KPP=IP(JPL,L)
    KLL=IW(JPL,L)+KPP-1
    IF (KPP.GT.KLL) GOTO 250
DO 246 K=KPP,KLL
    J=IND2(K)
    IF (L.EQ.2) J=IND1(K)
    IL=IW(J,4)
    IN=IW(J,5)
    IW(J,5)=-1
    IF (IN.LT.0) GOTO 246
    IF (IL.EQ.0) GOTO 240
    IW(IL,5)=IN
    GOTO 245
240  NZ=IW(J,1)+IW(J,2)+1
    IW(NZ,3)=IN
245  IF (IN.GT.0) IW(IN,4)=IL
246  CONTINUE
250  CONTINUE
C
C REMOVE JP(L1) FROM ORDERING VECTORS

```

BEST AVAILABLE COPY

```

      IL=IW(JPL,4)
      IN=IW(JPL,5)
      IW(JPL,5)=-1
      IF (IN.LT.0) GOTO 266
      IF (IL.EQ.0) GOTO 260
      IW(IL,5)=IN
      GOTO 265
260  NZ=IW(JPL,1)+IW(JPL,2)+1
      IW(NZ,3)=IN
265  IF(IN.GT.0)      IW(IN,4)=IL
266  CONTINUE
C
C  STORE PIVOT.
      DO 267 L=1,PP
267  IW(JP(L),4)=-IIP+1-L
C
C  CREATE A NEW ENTRY FOR ROW JP1 (ANDJP2) IF NECESSARY AND TRANSFORM
C  COLUMN PART TO ROWFILE.  REMOVE PIVOTAL ROW(S) FROM THE COLUMN FILE.
      DO 325 LL=1,PP
      JPL=JP(LL)
      NZC=IW(JPL,2)
      IF (NZC.EQ.0) GOTO 290
C
C  COMPRESS ROWFILE IF NECESSARY
      IF (NUEL+NZC+IW(JPL,1).LT.IA2) GOTO 270
      IF (LROW+NZC+IW(JPL,1).GT.IA2 .OR. MCP.GE.MCP) GOTO 670
      CALL COMPRE(A,IND2,IA2,N,IW,IP,.TRUE.)
270  KPP=IP(JPL,2)
      KLL=KPP+NZC-1
      IP(JPL,2)=NUEL+1
      DO 280 K=KPP,KLL
      NUEL=NUEL+1
      LCOL=LCOL-1
      I=IND1(K)
      KIP=IP(I,1)
      KIL=KIP+IW(I,1)-1
      DO 275 KK=KIP,KIL
      IF (IND2(KK).EQ.JPL) GOTO 276
275  CONTINUE
276  IND2(NUEL)=I
      A(NUEL)=A(KK)
      IND1(K)=0
C
C  MOVE ELEMENT FROM ROWFILE.
      KRL=IP(I,1)+IW(I,1)-1
C  MOVE ELEMENT FROM ROWFILE.
      KRL=IP(I,1)+IW(I,1)-1
      IF (KK.EQ.KRL) GOTO 279
      IL=IND2(KRL)
      IND2(KK)=IL
      A(KK)=A(KRL)
279  IND2(KRL)=0
280  IW(I,1)=IW(I,1)-1
C
C  MOVE ROWFILE OF ROW JP(ILL) IF NZC .GT. 0
290  KPP=IP(JPL,1)
      IF (NZC.GT.0) IP(JPL,1)=IP(JPL,2)

```

```

      KLL=IW(JPL,1)+KPP-1
      IF (KPP.GT.KLL) GOTO 320
      DO 319 K=KPP,KLL
      J=IND2(K)
      KPC=IP(J,2)
      NZ=IW(J,2)-1
      IW(J,2)=NZ
      KLC=KPC+NZ
      IF (KLC.GE.KPC) GOTO 314
      IND1(KPC)=0
      GOTO 317
314  DO 315 KC=KPC,KLC
      IF (JPL.EQ.IND1(KC)) GOTO 316
315  CONTINUE
316  IND1(KC)=IND1(KLC)
      IND1(KLC)=0
317  LCOL=LCOL-1
      IF (NZC.EQ.0) GOTO 319
      NUEL=NUEL+1
      A(NUEL)=A(K)
      IND2(NUEL)=IND2(K)
      IND2(K)=0
319  CONTINUE
C
C  UPDATE ROWPOINTERS TO ROW JP(LL)
320  IW(JPL,1)=IW(JPL,1)+NZC
325  IW(JPL,2)=-PP
C
C  MOVE THE PIVOTAL OFF DIAGONAL ELEMENT TO THE FRONT OF ROW JP1 IF PP=2
      NZC=IW(JP1,1)
      IF (PP.EQ.1) GOTO 360
      KPP=IP(JP1,1)
      KLL=KPP+NZC-1
      DO 326 K=KPP,KLL
      J=IND2(K)
      IF (J.EQ.JP2) GOTO 327
326  CONTINUE
327  IF (K.EQ.KPP) GOTO 328
      A1=A(KPP)
      I1=IND2(KPP)
      A(KPP)=A(K)
      IND2(KPP)=JP2
      A(K)=A1
      IND2(K)=I1
C
C  IF PP=2 THEN CONSTRUCT A PSEUDOROW CONTAINING THE UNION OF COLUMN
C  NUMBERS OF THE TWO PIVOTAL ROWS AT THE END OF THE ROWFILE.
C
C  COMPRESS ROWFILE IF THERE IS NOT ROOM FOR NEW ELEMENTS.
328  NZC=IW(JP1,1)+IW(JP2,1)-1
      IF (NUEL+NZC.LT.IA2) GOTO 329
      IF (LROW+NZC.GT. IA2 .OR. MCP.GE.MCP) GOTO 670
      CALL COMPRES(A,IND2,IA2,N,IW,IP,.TRUE.)
329  KP1=IP(JP1,1)+1
      KL1=IW(JP1,1)+KP1-2
      IF (KP1.GT.KL1) GOTO 331
      DO 330 K=KP1,KL1

```

BEST AVAILABLE COPY

```

330 IW(IND2(K),6)=1
331 NZC=IA2+1
      KP2=IP(JP2,1)
      KL2=IW(JP2,1)+KP2-1
      IF (KP2.GT.KL2) GOTO 341
      DO 340 K=KP2,KL2
      I=IND2(K)
      NZC=NZC-1
      IND2(NZC)=I
340 IW(I,6)=-1
341 IF (KP1.GT.KL1) GOTO 351
      DO 350 K=KP1,KL1
      I=IND2(K)
      IF (IW(I,6).EQ.-1) GOTO 350
      NZC=NZC-1
      IND2(NZC)=I
349 IW(I,6)=-1
350 CONTINUE
351 NZC=IA2-NZC+1

```

C
C PERFORM THE ELIMINATION LOOPING ON NONZEROS IN PIVOT COLUMN(S).

```

360 IF (NZC.EQ.0) GOTO 485
      DO 363 L=1,PP
      KP(L)=IP(JP(L),1)
      KL(L)=KP(L)+IW(JP(L),1)-1
      IF (PP.EQ.2.AND.L.EQ.1) KP(L)=KP(L)+1
      IF (KP(L).GT.KL(L)) GOTO 363

```

C
C UNPACK PIVOT ROW(S) IN IW(*,5+L).

```

      KPP=KP(L)
      KLL=KL(L)
      W(JP(L))=0.
      DO 362 K=KPP,KLL
      J=IND2(K)
      W(J)=0.
362 IW(J,5+L)=K-KPP
363 CONTINUE
      DO 480 NC=1,NZC
      KC=IP(JP1,1)+NC-1
      IF (PP.EQ.2) KC=IA2-NZC+NC
      IR=IND2(KC)
      IF (PP.EQ.1) AL=A(KC)/D(JP1)

```

C
C COMPRESS ROWFILE UNLESS IT IS CERTAIN THAT THERE IS ROOM FOR NEW ROW.

```

      IF (NUEL+IW(IR,1)+NZC.LE.IA2-NZC) GOTO 365
      IF (NCP.GE.MCP .OR. LROW+IW(IR,1)+NZC.GT.IA2-NZC) GOTO 670
      NZ0=NZC
      IF (PP.EQ.1) NZ0=0
      CALL COMPRE(A,IND2,IA2-NZ0,N,IW,IP,.TRUE.)
      DO 364 L=1,PP
      KP(L)=IP(JP(L),1)
364 KL(L)=KP(L)+IW(JP(L),1)-1
      IF (PP.EQ.2) KP(1)=KP(1)+1
365 KR=IP(IR,1)
      KRL=KR+IW(IR,1)-1
      IP(IR,1)=NUEL+1
      IF (PP.EQ.1) GOTO 377

```

```

AIR1=0.
AIR2=0.
IWI1=IW(IR,6)
IWI2=IW(IR,7)
IF (IWI1.GE.0) AIR1=A(KP(1)+IWI1)
IF (IWI2.GE.0) AIR2=A(KP(2)+IWI2)

```

C

C TRANSFER MODIFIED ELEMENTS.

```

377 IF (KR.GT.KRL) GOTO 420
DO 390 KS=KR,KRL
J=IND2(KS)
AJR1=0.
IWJ1=IW(J,6)
IW(J,6)=-IWJ1-2
IF (IWJ1.GE.0) AJR1=A(KP(1)+IWJ1)
IF (PP.EQ.1) GOTO 380
AJR2=0.
IWJ2=IW(J,7)
IW(J,7)=-IWJ2-2
IF (IWJ2.GE.0) AJR2=A(KP(2)+IWJ2)
A1=A(KS)-AIR1*AJR1*A22+AIR2*AJR1*A12-AIR2*AJR2*A11+AIR1*AJR2*A12
GOTO 381
380 A1=A(KS)-AL*AJR1
381 IND2(KS)=0
* GI=DABS(A1)
  GI=ABS(A1)
  IF (W(IR).LT.GI) W(IR)=GI
  IF (W(J).LT.GI) W(J)=GI
* G=DMAX1(G,GI)
  G=AMAX1(G,GI)
  NUEL=NUEL+1
  A(NUEL)=A1
390 IND2(NUEL)=J

```

C

C SCAN PIVOT ROW FOR FILLS.

```

420 DO 480 NC1=1,MZC
  IF (PP.EQ.1) GOTO 421
  KS=IA2+1-NC1
  GOTO 422
421 KS=KP(1)-1+NC1
422 J=IND2(KS)
  AJR1=0.
  IWJ1=IW(J,6)
  AJR2=0.
  IWJ2=IW(J,7)
  IF (IWJ1.LE.-1 .AND. IWJ2.LE.-1 .OR. J.GT.IR) GOTO 469
  IF (IWJ1.GE.0) AJR1=A(KP(1)+IWJ1)
  IF (PP.EQ.1) GOTO 425
  IF (IWJ2.GE.0) AJR2=A(KP(2)+IWJ2)
  A1=-AIR1*AJR1*A22+AIR2*AJR1*A12-AIR2*AJR2*A11+AIR1*AJR2*A12
  GOTO 425
425 A1=-AL*AJR1
426 IF (IR.NE.J) GOTO 429
  D(IR)=D(IR)+A1
  A1=D(IR)
* G=DMAX1(DABS(A1),G)
  G=AMAX1(ABS(A1),G)

```

BEST AVAILABLE COPY

```

      GOTO 469
429  NUEL=NUEL+1
      LROW=LROW+1
      A(NUEL)=A1
      IND2(NUEL)=J
*    GI=DABS(A1)
      GI=ABS(A1)
      IF (W(IR).LT.GI) W(IR)=GI
      IF (W(J).LT.GI) W(J)=GI
C
C CREATE FILL IN COLUMN FILE.
      NZ=IN(J,2)
      K=IP(J,2)
      KL1=K+NZ-1
C
C IF POSSIBLE PLACE NEW ELEMENT AT THE END OF PRESENT ENTRY.
      IF (KL1.NE.NUCL) GOTO 430
      IF (LCOL.GE.IA1) GOTO 440
      NUCL=NUCL+1
      GOTO 435
430  IF (IND1(KL1+1).NE.0) GOTO 440
435  IND1(KL1+1)=IR
      LCOL=LCOL+1
      GOTO 465
C
C NEW ENTRY HAS TO BE CREATED.
440  IF (NUCL+NZ+1.LT.IA1) GOTO 450
C
C COMPRESS COLUMNFILE IF THERE IS NOT ROOM FOR NEW ENTRY.
      IF (NCP.GE.NCP .OR. LCOL+NZ+1.GE.IA1) GOTO 602
      CALL COMPRES(A,IND1,IA1,N,IN(1,2),IP(1,2),.FALSE.)
      K=IP(J,2)
      KL1=K+NZ-1
C
C TRANSFER OLD ENTRY INTO NEW.
450  IP(J,2)=NUCL+1
      IF (K.GT.KL1) GOTO 461
      DO 460 KK=K,KL1
      NUCL=NUCL+1
      IND1(NUCL)=IND1(KK)
460  IND1(KK)=0
C
C ADD THE NEW ELEMENT.
461  NUCL=NUCL+1
      IND1(NUCL)=IR
      LCOL=LCOL+1
*465  G=DNAX1(G,GI)
465  G=ANAX1(G,GI)
      IN(J,2)=NZ+1
469  IF (IN(J,7).LT.-1) IN(J,7)=-IN(J,7)-2
      IF (IN(J,6).LT.-1) IN(J,6)=-IN(J,6)-2
      IN(IR,1)=NUEL+1-IP(IR,1)
480  CONTINUE
C
C CLEAN IN(*,6) AND IN(*,7) BY SCANNING THE PIVOT ROWS
      DO 484 NC=1,NZC
      IF(PP.EQ.1) GOTO 481

```

```

      K=IA2+1-NC
      GOTO 482
481  K=KP(1)+NC-1
      A(K)=A(K)/D(JP1)
482  J=IND2(K)
      DO 484 L=1,PP
484  IW(J,5+L)=-1
C
C INSERT ROWS/COLUMNS INVOLVED IN ELIMINATION IN LINKED LISTS OF EQUAL
C NUMBER OF NON ZEROS
485  K1=IP(JP1,1)
      IF (PP.EQ.2) K1=IA2-NZC+1
      K2=K1+NZC-1
      IF (K1.GT.K2) GOTO 491
      DO 490 K=K1,K2
      IR=IND2(K)
      NZ=IW(IR,1)+IW(IR,2)+1
      IN=IW(NZ,3)
      IW(IR,5)=IN
      IW(IR,4)=0
      IW(NZ,3)=IR
490  IF (IN.NE.0) IW(IN,4)=IR
491  IF (PP.EQ.1) GOTO 500
      D(JP1)=A22
      D(JP2)=A11
      A(IP(JP1,1))=-A12
500  CONTINUE
C ELIMINATION LOOP TERMINATES.
C
C MAKE AN ORDERD LIST OF PIVOTS.
      DO 510 I=1,N
      IW(I,2)=-IW(I,2)
      IR=-IW(I,4)
510  IW(IR,3)=I
      G=GG
      GOTO 720
C
C THE FOLLOWING INSTRUCTIONS IMPLEMENT THE FAILURE EXITS.
610  IF (FIL.GT.0) WRITE(FIL,620) K,I,J
620  FORMAT(/34X,7HELEMENT,I7,10H IS IN ROW,I5,11H AND COLUMN,I5)
      G=-1
      GOTO 700
630  IF (FIL.GT.0) WRITE(FIL,640) I
640  FORMAT(/34X,5HROW ,I5,16H HAS NO ELEMENTS)
      G=-2
      GOTO 700
650  IF (FIL.GT.0) WRITE(FIL,660) IR,J
660  FORMAT(/34X,35HTHERE IS MORE THAN ONE ENTRY IN ROW,I5,
      * 11H AND COLUMN,I5)
      G=-3
      GOTO 700
670  IF (FIL.GT.0) WRITE(FIL,680)
680  FORMAT(/34X,16HIA2 IS TOO SMALL)
      G=-4
      GOTO 700
682  IF (FIL.GT.0) WRITE(FIL,683)
683  FORMAT(/34X,16HIA1 IS TOO SMALL)

```

G=-5
700 IF (FIL.GT.0) WRITE(FIL,710)
710 FORMAT(33H+ERROR RETURN FROM INDAML BECAUSE)
720 RETURN
END

BEST AVAILABLE COPY

```

SUBROUTINE INDOPR (A,IND2,IA2,N,IW,IP,O,B,G)
* IMPLICIT REAL*8 (A-H,S-Z),INTEGER (I-R)
IMPLICIT REAL (A-H,S-Z),INTEGER (I-R)
INTEGER IP(N,2),KP(2),KL(2)
* INTEGER*2 IND2(IA2),IW(N,5)
INTEGER IND2(IA2),IW(N,5)
* DOUBLE PRECISION A(IA2),O(N),B(N)
DIMENSION A(IA2),O(N),B(N)
C INDOPR PERFORMS THE SOLUTION OF AX = B WHEN A HAS BEEN THROUGH INDANL.
C
IF (G.LT.0) GOTO 250
DO 130 IIP=1,N
IC1=IW(IIP,3)
PP=IW(IC1,2)
IF (PP.EQ.0) GOTO 130
KP(1)=IP(IC1,1)
KL(1)=KP(1)+IW(IC1,1)-1
B1=B(IC1)
IF (PP.EQ.1) GOTO 100
IC2=IW(IIP+1,3)
KP(2)=IP(IC2,1)
KL(2)=KP(2)+IW(IC2,1)-1
B2=B(IC2)
IW(IC2,2)=0
A12=A(KP(1))
KP(1)=KP(1)+1
B(IC1)=O(IC1)*B1+A12*B2
B(IC2)=A12*B1+O(IC2)*B2
100 DO 120 LL=1,PP
KPP=KP(LL)
KLL=KL(LL)
IC=IC1
IF (LL.EQ.2) IC=IC2
IF (KPP.GT.KLL) GOTO 120
DO 110 K=KPP,KLL
IR=IND2(K)
110 B(IR)=B(IR)-A(K)*B(IC)
120 CONTINUE
130 CONTINUE
WRITE(6,1000) KLL
1000 FORMAT(1X,5H****,I10)
DO 200 IPI=1,N
IIP=N+1-IPI
IR2=IW(IIP,3)
PP=IW(IR2,2)
BIR=0.
KPP=IP(IR2,1)
KLL=KPP+IW(IR2,1)-1
IF (PP.EQ.2) KPP=KPP+1
IF (KPP.GT.KLL) GOTO 150
DO 149 K=KPP,KLL
IC=IND2(K)
149 BIR=BIR-A(K)*B(IC)
150 IF (PP.EQ.1) B(IR2)=B(IR2)/O(IR2)+BIR
IF (PP.EQ.0) GOTO 160
IF (PP.NE.2) GOTO 200
A12=A(KPP-1)

```

BEST AVAILABLE COPY

```

IR3=IN(IIP+1,3)
B(IR2)=B(IR2)+B(IR2)*B IR
B(IR3)=B(IR3)+A12*BIR
GOTO 200
160 IR1=IN(IIP-1,3)
B(IR2)=B(IR2)+D(IR2)*B IR
B(IR1)=B(IR1)+A(IP(IR1,1))*BIR
200 CONTINUE
250 RETURN
END

```

BEST AVAILABLE COPY

```

      SUBROUTINE COMPRE(A,IRN,IA,N,IW,IP,ROW)
C
C THIS SUBROUTINE IS IDENTICAL TO THE HARNELL SUBROUTINE LA05ED.
C   DOUBLE PRECISION A(IA)
C   DIMENSION      A(IA)
*   INTEGER*4 IP(N)
C   INTEGER      IP(N)
*   INTEGER*2 IRN(IA),IW(N)
C   INTEGER      IRN(IA),IW(N)
C   LOGICAL ROW
C   COMMON/INDEFI/FIL,NUEL,LROW,LCOL,NCP,NUCL
C   NCP=NCP+1
C   DO 5 J=1,N
C
C STORE LAST ELEMENT OF ENTRY IN IW(J). THEN OVERWRITE IT BY -J
      NZ=IW(J)
      IF (NZ.LE.0) GOTO 5
      K=IP(J)+NZ-1
      IW(J)=IRN(K)
      IRN(K)=-J
5     CONTINUE
C
C KN IS POSITION OF NEXT ENTRY IN COMPRESSED FILE.
      KN=0
      IPI=0
      KL=NUCL
      IF (ROW) KL=NUEL
C
C LOOP THROUGH OLD FILE SKIPPING ZERO ELEMENTS AND MOVING GENUINE
C ELEMENTS FORWARD. THE ENTRY NUMBER BECOMES KNOWN ONLY WHEN
C ITS END IS DETECTED BY PRESENCE OF A NEGATIVE INTEGER.
      DO 25 K=1,KL
      IF (IRN(K).EQ.0) GOTO 25
      KN=KN+1
      IF(ROW) A(KN)=A(K)
      IF (IRN(K).GE.0) GOTO 20
C
C END OF ENTRY. RESTORE IRN(K), SET POINTERS TO START OF ENTRY AND
C STORE CURRENT KN IN IPI READY FOR USE WHEN NEXT LAST ENTRY IS
C DETECTED.
      J=-IRN(K)
      IRN(K)=IW(J)
      IP(J)=IPI+1
      IW(J)=KN-IPI
      IPI=KN
20    IRN(KN)=IRN(K)
25    CONTINUE
      IF (ROW) GOTO 26
      NUCL=KN
      GOTO 27
26    NUEL=KN
27    RETURN
      END

```

BEST AVAILABLE COPY

BLOCK DATA
COMMON/INDEFI/FIL,MUEL,LROW,LCOL,NCP,NUCL
INTEGER FIL
DATA FIL/6/
END

REFERENCES

1. "The NASTRAN Theoretical Manual," Edited by R.H. MacNeal, National Aeronautics and Space Administration, Washington, D.C., (1969), p. 3.1-1.
2. Eisenstat, S.C. et al., "Yale Sparse Matrix Package I. The Symmetric Codes," Yale University Dept. of Computer Science Report No. 112, New Haven, Connecticut.
3. Eisenstat, S.C. et al., "Yale Sparse Matrix Package II. The Nonsymmetric Codes," Yale University Dept. of Computer Science Report No. 114, New Haven, Connecticut.
4. Munksgaard, N., "Fortran Subroutines for Direct Solution of Sets of Sparse and Symmetric Linear Equations," Institute for Numerical Analysis Rept. No. 77-05, The Technical Univ. of Denmark, 2800 Lyngby, Denmark (Apr 1977).
5. Duff, I.S. et al., "Direct Solution of Sets of Linear Equations Whose Matrix Is Sparse, Symmetric, and Indefinite," Atomic Energy Research Establishment Report CSS 44, Harwell, Dideot, Oxon, United Kingdon (Jan 1977).
6. Rheinboldt, W.C. and C.K. Mesztenyi, "Programs for the Solution of Large Sparse Matrix Problems Based on the Arc-Graph Structure," University of Maryland Computer Science Center Technical Rept. TR-262, College Park, MD (Sep 1973).
7. Mesztenyi, C.K. and W.C. Rheinboldt, "Further Programs for the Solution of Large Sparse Systems of Linear Equations," University of Maryland Computer Science Center Technical Rept. TR-394, College Park, MD (Aug 1975).
8. Gignac, D.A., "CSKYDG2: An Out-of-Core Cholesky Algorithm Equation Solver (with Respect to Profile) for Large Positive Definite Systems of Linear Equations," Naval Ship Research and Development Center Rept. 4655 (Mar 1975).